(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) N° de publication : **2 752 966**
(à n'utiliser que pour les commandes de reproduction)

(21) N° d'enregistrement national : **97 10435**

(51) Int Cl[6] : **G 06 F 15/16**

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt** : 18.08.97.

(30) **Priorité** : 19.08.96 US 699280.

(43) **Date de la mise à disposition du public de la demande :** 06.03.98 Bulletin 98/10.

(56) **Liste des documents cités dans le rapport de recherche préliminaire :** *Ce dernier n'a pas été établi à la date de publication de la demande.*

(60) **Références à d'autres documents nationaux apparentés :**

(71) **Demandeur(s) :** *SAMSUNG ELECTRONICS CO LTD* — KR.

(72) **Inventeur(s) :**

(73) **Titulaire(s) :** .

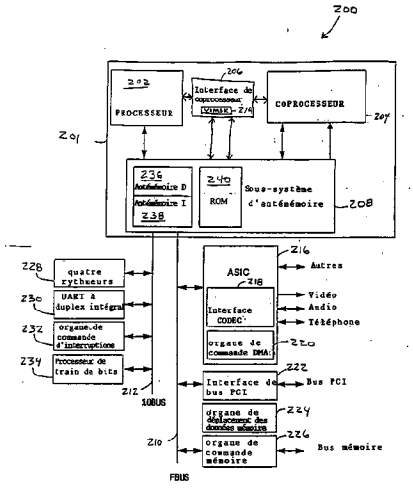
(74) **Mandataire :** CABINET WEINSTEIN.

(54) SAUVEGARDE ET RESTAURATION EFFICACES DE CONTEXTE DANS UN ENVIRONNEMENT A SYSTEME DE CALCUL MULTITACHE.

(57) La présente invention concerne un environnement à système de calcul multitâche.

Le processeur (202) demande à un coprocesseur (204) de commuter le contexte approprié pour déterminer le programme d'exécution, le coprocesseur (204) répond en arrêtant l'exécution du programme et en ne sauvegardant que la quantité minimale d'informations d'état de processeur nécessaire pour une restauration avec succès du programme. Le point approprié est choisi par le programmeur d'application à en emplacement du programme d'exécution qui exige de préserver une partie minimale d'informations de processeur aux bornes de la commutation de contexte.

Et ne sauvegardant qu'une quantité minimale d'informations de processeur, on accumule des économies de temps de processeur aux bornes des opérations de sauvegarde et de restauration de contexte.



FR 2 752 966 - A1

La présente invention concerne des systèmes de calcul et plus particulièrement des opérations de commutation de contexte dans un environnement multi-tache ayant par exemple une architecture à multiprocesseurs.

Un environnement multi-tache implique de manière générale l'exécution concomitante de plusieurs programmes dans un ordinateur, par exemple, une exécution de programmes à découpage du temps. Les utilisateurs de l'ordinateur ont l'impression que les programmes sont exécutés en parallèle alors qu'en réalité l'ordinateur commute entre les programmes. Lorsque l'exécution d'un programme est à découpage du temps, un programme est exécuté pendant une période de temps avant d'être "à commutation de contexte" (commutation de tache) pour exécuter un autre programme.

Lors de la reprise d'un programme à commutation de contexte, le programme devrait reprendre l'exécution à l'emplacement précis où l'exécution a été préalablement terminée. En anticipation de la reprise ultérieure du programme à commutation de contexte, un processeur effectue un processus de sauvegarde de l'état du programme qui était à commutation de contexte avant le chargement et l'exécution d'un autre programme par le processeur. Cet état de programme est représenté dans divers emplacements mémoire par l'état du processeur lorsque le programme est à commutation de contexte.

La commutation de contexte est transparente au programmeur du programme à commutation de contexte parce que, du point de vue du programmeur, la commutation de contexte se produit à des temps arbitraires et ainsi à des emplacements arbitraires du programme. Supplémentairement, le système de fonctionnement qui met en oeuvre la commutation de contexte ne connaît pas l'état du programme dont le contexte a été commuté. Il en résulte que toutes les informations d'état du processeur doivent être sauvegardées lorsque le programme est commuté en contexte. Les informations d'état du processeur

comprennent tous les registres visibles d'architecture et de logiciel ainsi que tous emplacements mémoire qui sont établis à des adresses connues seulement à l'intérieur du processeur telles qu'une mémoire de travail.

Pour une architecture de processeur présentant une grande quantité d'informations d'état, le procédé traditionnel de commutation de contexte de programme impliquant la sauvegarde et la restauration de toutes les informations d'état du processeur est inefficace et peut avoir un impact négatif sur les performances du processeur. L'inefficacité et l'impact de performance négatif associé à la commutation de contexte doit être attribué en partie à des opérations de processeur qui nécessitent d'accéder à des emplacements de mémoire multiples affectés à la mise en mémoire des informations d'état du processeur et à des opérations qui enregistrent les informations d'état du processeur aux bornes de bus de relativement faible largeur de bande vers d'autres emplacements mémoire. Cette même inefficacité de la commutation de contexte de type conditionnel a également un impact négatif sur les performances du processeur au cours du processus de restauration de programme lorsque l'ensemble des informations d'état de processeur mises en méoire est transféré en retour vers les emplacements mémoire affectés à l'état des informations de processeur. Cette inefficacité est accumulée au cours de chaque commutation de contexte de programme. Par exemple, une architecture à processeur de signaux multimédia peut avoir un état de processeur représenté par plus de 7 kilo-octets d'informations dans plus de cent (100) registres et emplacements mémoire. Chaque commutation de contexte nécessite de manière traditionnelle de transférer l'ensemble de ces informations vers un emplacement de mise en mémoire d'état de processeur.

La réduction des impacts négatifs de performances qui sont associés à la commutation de contexte dans un système multi-tache, tel qu'un système multimédia

employant un processeur de signaux multimédia, réduit de manière avantageuse les retards indésirables entre l'exécution des programmes. Ceci peut être particulièrement sensible lorsque des programmes sont impliqués dans le traitement de grandes quantités d'informations tel que dans un environnement de traitement multimédia.

La présente invention réduit de manière avantageuse la quantité de temps de processeur nécessaire pour la commutation de contexte entre programmes. Certains points dans un programme au cours de commutation de contexte nécessitent une mise en mémoire de plus d'informations d'état de processeur de manière à reprendre avec succès l'exécution du programme plus tard que d'autres points. Dans un mode de réalisation de la présente invention, une architecture à multiprocesseurs permet à un processeur de requérir l'exécution d'un programme en cours par un processeur pour s'interrompre lui-même à un point approprié en anticipation de la commutation de contexte du programme en cours. Le point approprié correspond à un point du programme en cours d'exécution qui peut réduire la quantité d'informations d'état de processeur nécessaire, pour reprendre avec succès le programme de commutation de contexte et peut ne pas suivre immédiatement la réception de la requête de commutation de contexte.

Dans un autre mode de réalisation de l'invention, les responsabilités pour les fonctions de sauvegarde et de restauration de contexte dans un environnement multi-tache sont attribuées entre un processeur et le programme d'application en cours d'exécution. Dans ce mode de réalisation, le programme est marqué, par exemple avec une inscription de programme de commutation de contexte conditionnel, dans des emplacements multiples correspondant à des points du programme requérant une quantité réduite d'informations d'état de processeur pour reprendre avec succès l'exécution du programme. Lorsque

le programme atteint l'emplacement marqué et qu'une requête de commutation de contexte a été reçue et détectée, le programme déclenche une réponse pour mettre en mémoire seulement les informations d'état de processeur nécessaires pour une reprise avec succès du programme. Le programme devrait être marqué de manière suffisamment fréquente pour éviter tout retard notable entre l'excitation d'une requête de commutation de contexte et une commutation de contexte en réponse.

Dans un autre mode de réalisation, la présente invention est un procédé comprenant les opérations consistant à exécuter un premier programme sur un premier processeur, à recevoir une requête de commutation de contexte d'un second processeur, et à répondre à la requête de commutation de contexte au niveau d'un point approprié du premier programme, dans lequel le point approprié est associé à un repère dans le programme qui indique un point approximatif du premier programme requérant une quantité minimum de mise en mémoire d'informations d'état de processeur en vue d'une restauration avec succès du programme.

Dans un autre mode de réalisation, la présente invention est un dispositif de calcul comprenant un premier processeur dans un environnement multi-tache pour exécuter des programmes présentant des ensembles respectifs de repères de commutation de contexte intercalés, et une première mémoire, couplée au premier processeur, affectée au stockage des informations d'état de processeur. Le dispositif de calcul comprend en outre une seconde mémoire couplée au premier processeur, un détecteur de requête de commutation de contexte fonctionnant sur le premier processeur pour détecter, après que le processeur rencontre l'un des repères dans un programme d'exécution, une requête pour commuter le contexte du programme, et un module de sauvegarde de contexte fonctionnant sur le premier processeur pour répondre à une requête de commutation de contexte

détectée en sauvegardant dans la seconde mémoire les informations d'état de processeur placées dans la première mémoire.

Diverses autres caractéristiques de l'invention ressortent d'ailleurs de la description détaillée qui suit.

Des formes de réalisation de l'objet de l'invention sont représentées, à titre d'exemples non limitatifs, aux dessins annexés.

La figure 1 est un schéma synoptique de type évolué illustrant un système à multiprocesseurs multimédia conforme à un mode de réalisation de la présente invention.

La figure 2 est un schéma synoptique montrant un processeur de signaux multimédia inclu dans le dispositif à multiprocesseurs multimédia illustré à la figure 1.

La figure 3 est un schéma synoptique montrant un coprocesseur du processeur de signaux multimédia illustré à la figure 2.

La figure 4 est un schéma synoptique montrant des chemins de données d'exécution de coprocesseur du coprocesseur illustré à la figure 3.

La figure 5 est un schéma synoptique illustrant une architecture de micro-programmation du processeur de signaux multimédia.

La figure 6 est un organigramme qui illustre des opérations concernant une sauvegarde de contexte de programme efficace ainsi qu'une restauration de programme efficace dans une architecture à multiprocesseurs.

La figure 7 est une illustration d'une partie d'un déroulement de programme pouvant être exécuté et présentant des instructions de commutation conditionnelle de contexte intercalées.

En se référant à la figure 1, un schéma synoptique de type évolué illustre un dispositif à multiprocesseurs multimédia 100 comprenant un processeur central 102 et un processeur de signaux multimédia 200. Un processeur

central typique 102 est un processeur x86 tel qu'un processeur Pentium™ ou Pentium Pro™ d'Intel Corporation. Le processeur central 102 exécute des programmes basés sur des instructions et des données maintenues principalement dans une mémoire système 104 et une antémémoire 105. Le processeur central 102 communique avec le processeur de signaux multimédia 200 par l'intermédiaire d'un jeu de puces PC 107 et d'un bus système 106, tel qu'un bus PCI. Le proceseur de signaux multimédia 200 fournit une interface pour divers blocs fonctionnels tels qu'un CODEC audio et de communication 108 pour recevoir une communication audio et une communication téléphonique, un convertisseur A/N vidéo 110 pour recevoir des signaux d'entrée vidéo, un convertisseur N/A vidéo 112 pour émettre des signaux de sortie vidéo, et une mémoire tampon SDRAM de trame 114. Dans un mode de réalisation, le processeur de signaux multimédia est l'un quelconque des processeurs de signaux multimédia de la famille MSP de Samsung Semiconducteur Inc (MSP de Samsung).

En se référant à la figure 2, un schéma synoptique montre le processeur de signaux multimédia 200 du dispositif à multiprocesseurs multimédia 100 (figure 1). Le processeur de signaux multimédia 200 comprend une mémoire centrale de processeur de signaux numériques (PSN) 201 qui fournit une interface par l'intermédiaire d'un bus rapide (FBUS) 210 pour un ensemble de périphériques FBUS comprenant, par exemple, une interface de bus PCI à 32 bits 222, un organe de commande de mémoire SDRAM à 64 bits 226, un organe de commande DMA à 8 canaux 220, un bloc logique ASIC 216, et un organe de déplacement de données mémoire 224 pour déplacer des données entre le processeur central 102 et la mémoire tampon SDRAM de trame 114. L'interface de bus PCI 222 fournit une interface pour le bus système 106 et fonctionne, par exemple, à 33 MHz. Le bloc logique ASIC 216 fournit une logique de commande pour mettre

éventuellement en oeuvre une fonctionnabilité adaptée au besoin particulier de l'utilisateur. Le bloc logique ASIC 216, dans un mode de réalisation, fournit 10 kiloportes comprenant des interfaces vers divers CODEC analogiques et dispositifs d'entrée/sortie (E/S) spécifiques aux utilisateurs. L'organe de déplacement de données mémoire 224 transfère des données DMA du processeur central 102 vers la mémoire SDRAM 114 qui est locale au processeur de signaux multimédia 200. La mémoire centrale de PSN 201 fournit également une interface par l'intermédiaire d'un bus E/S pour un ensemble de dispositifs à bus E/S comprenant, par exemple, un synchronisateur d'intervalle programmable ou rythmeur 228 à compatibilité 8254, une ligne série UART 230 à compatibilité 16450, un organe de commande d'interruption programmable 232 à compatibilité 8259, et un processeur de train de bits 234 pour traiter un train de bits vidéo. Pour plus de renseignements concernant le processeur de train de bits 234 il y a lieu de se référer aux demandes de brevet US déposées le 19 Août 1996 sous les N° 08/699 303 et 699 382 au nom de C. Reader et al intitulés "Methods and Apparatus for Processing Video Data" qui sont incorporées aux présentes par référence à leur texte complet.

La mémoire centrale de PSN 201 est le moteur de calcul du processeur de signaux multimédia 200 et comprend un processeur 202, un coprocesseur 204, un sous-système d'antémémoire 208, le bus rapide (FBUS) 210 et le bus E/S 212. Dans un mode de réalisation, le proceseur 202 est un processeur de commande RISC à 32 bits de type ARM7™ qui effectue des fonctions de traitement générales telles que des requêtes d'autorisation de commutation de contexte, des opérations de système de fonctionnement en temps réel, un traitement d'interruption et d'exception, une gestion de dispositifs entrée/sortie, une communication avec le processeur central 102, et analogues. Dans un mode de réalisation, le processeur 202 fonctionne à 40 MHz. Le processeur 202 fournit une

interface pour le coprocesseur 204 par l'intermédiaire d'une interface de coprocesseur 206.

Le processeur 202 effectue le traitement d'exceptions en réponse à des exceptions, en général des conditions qui se produisent au cours du traitement d'instructions, en provoquant une modification du déroulement de la commande d'exécution. Pour plus de renseignements concernant le traitement d'exception, il y a lieu de se référer aux demandes de brevets US déposées aux noms de Song et al le 19 Août 1996 l'une sous le N° 08/699 295, intitulée "System And Method For Handling Software Interrupts With Argument Passing", et l'autre sous le N° 08/699 294, intitulée "System And Method For Handling Interrupt And Exception Events In An Asymmetric Multiprocessor Architecture", qui sont incorporées aux présentes par référence à leur texte complet.

Le coprocesseur 204 est le moteur de traitement de signaux numériques du processeur de signaux multimédia 200. Dans un mode de réalisation, le coprocesseur 204 est un processeur vectoriel de la famille MSP de Samsung. En tant que processeur vectoriel, le coprocesseur 204 présente une architecture Instruction Unique-Données Multiples et comprend un moteur RISC pipeliné qui fonctionne sur des éléments de données multiples en parallèle pour effectuer des fonctions de traitement de signaux telles que des Transformés de Cosinus Discrètes (TCD), un filtrage FIR, une convolution, une estimation de mouvement vidéo et d'autres opérations de traitement. Le coprocesseur 204 supporte une arithmétique vectorielle dans laquelle des éléments de données multiples sont actionnés en parallèle, à la manière d'un processus vectoriel, par un ensemble d'unités d'exécution vectorielles. Le coprocesseur 204 exécute à la fois des opérations scalaires et des opérations vectorielles et scalaires combinées. Les éléments de données multiples du coprocesseur 204 sont comprimés dans un vecteur à 576 bits qui est calculé à un débit de trente deux opérations

arithmétique à virgule fixe de 8/9 bits, seize opérations arithmétiques à virgule fixe de 16 bits, ou huit opérations arithmétiques à virgule fixe ou à virgule flottante par cycle (par exemple 12,5 ns). La plupart des opérations scalaires à 32 bits sont pipelinées à une cadence d'une instruction par cycle, tandis que la plupart des opérations vectorielles à 576 bits sont pipelinées à une cadence d'une instruction en deux cycles. Les opérations de charge et de mise en mémoire présentent une simultanéité d'exécution avec les opérations arithmétiques et sont exécutées de manière indépendante par des circuits séparés de charge et de mise en mémoire.

En se référant à la figure 3, le coprocesseur 204 présente quatre blocs fonctionnels comprenant une unité d'extraction d'instructions 302, un bloc décodeur et délivreur d'instructions 304, un chemin de données d'exécution d'instructions 306 et une unité de charge et de mise en mémoire 308. L'unité d'extraction d'instructions 302 et le bloc décodeur et délivreur d'instructions 304 sont inclus dans le coprocesseur 204 pour permettre au coprocesseur 204 de fonctionner de manière indépendante du processeur 202.

L'unité d'extraction d'instructions 302 effectue une extraction préalable des instructions et traite les instructions de déroulement de commande telles que des instructions de Branchement et de Sauts vers Sous-programmes. L'unité d'extraction d'instructions 302 contient une file d'attente à 16 entrées d'instructions d'extraction préalable pour le train d'exécution en cours et une file d'attente à huit entrées d'instructions d'extraction préalable pour le train de cible de Branchement. L'unité d'extraction d'instructions 302 reçoit, aux bornes d'un bus de largeur 256 bits, jusqu'à huit instructions provenant de l'antémémoire d'instructions dans un cycle. Le bloc décodeur et délivreur d'instructions 304 décode et organise toutes

les instructions exécutées par le coprocesseur 204. Le décodeur traite une instruction dans un cycle dans l'ordre de réception à partir de l'unité d'extraction d'instructions 302, tandis que le délivreur organise la plupart des instructions en dérangement dépendant à la fois de la ressource d'excution et de la disponibilité des données d'opérandes.

En se référant à la figure 4, le chemin de données d'exécution d'instructions 306 comprend un fichier registre à quatre entrées 402, huit multiplicateurs parallèles 32x32 404 et huit ALU à 36 bits 406. Le fichier registre 402 prend en charge deux opérations de lecture et deux opérations d'enregistrement par cycle. Les multiplicateurs parallèles 404 produisent jusqu'à huit multiplications à 32 bits dans un format entier ou à virgule flottante, ou seize multiplications à 16 bits ou trente deux multiplications à 8 bits par cycle. Les ALU 406 exécutent soit huit opérations ALU à 36 bits dans un format entier ou à virgule flottante, seize opérations ALU à 16 bits, ou trente deux opérations à 8 bits par cycle (par exemple 12,5 ns).

Le fichier registre 402 inclut un ensemble de registres spécialisés et un ensemble de registres d'adresses de retour. Les registres spécialisés comprennent un registre de commande vectorielle et d'état (VCSR), un compteur de programme vectoriel (VPC), un compteur de programme d'exception vectoriel (VEPC), un registre de sources d'interruption vectorielle (VISRC), un registre de synchronisation du processeur vectoriel et de commande (VASYNC) et d'autres registres tels que divers registres de comptage, de masquage, de dépassement de capacité et de point de rupture. Le compteur de programme vectoriel (VPC) constitue l'adresse de l'instruction suivante à exécuter par le processeur vectoriel 204.

Le registre de sources d'interruption vectorielle (VISRC) indique les sources d'interruption vers le

processeur 202. Des bits appropriés du VISRC sont réglés par le matériel lors de la détection d'exceptions. Les bits sont remis à l'état initial par le logiciel avant que reprenne l'exécution du coprocesseur 204. Tout bit réglé dans le VISRC amène le coprocesseur 204 à entrer dans l'état inactif (VP_IDLE). Si le bit d'autorisation d'interruption correspondant est réglé dans un registre VIMSK de l'interface de coprocesseur 206, une interruption IRQ est signalée au processeur 202.

Le coprocesseur 204 détecte des conditions d'exception, comprenant des exceptions précises et des exceptions imprécises. Les exceptions précises sont détectées par le coprocesseur 204 et rapportées avant l'instruction en défaut. Les exceptions précises comprennent une exception de point de rupture d'adresses d'instruction, une exception de point de rupture d'adresses de données, une exception d'instruction incorrecte, une exception d'un seul chemin, une exception de dépassement de capacité d'empilage d'adresses de retour, une exception de dépassement de capacité négative d'empilage d'adresses de retour, une exception VCINT, et une exception VCJOIN. Les exceptions imprécises du coprocesseur 204 sont détectées et rapportées après l'exécution d'un nombre variable d'instructions qui sont plus tard dans l'ordre programme de l'instruction en défaut. Les exceptions imprécises comprennent une exception d'adresses d'instructions incorrectes, une exception d'adresses de données incorrectes, une exception d'accès de données non alignées, une exception de dépassement de capacité entière, une exception de dépassement de capacité à virgule flottante, une exception d'opérandes incorrectes à virgule flottante, une exception de division par zéro à virgule flottante et une exception de division par zéro entière.

Le registre d'instruction d'interruption vectorielle (VIINS) est mis à jour avec l'instruction VCINT ou l'instruction VCJOIN du coprocesseur lorsque

l'instruction est exécutée pour interrompre le processeur 202.

Le processeur 202 déclenche les opérations du coprocesseur 204. Pour davantage de renseignements concernant les opérations de déclenchement par le processeur 202 du coprocesseur 204, il y a lieu de se référer aux demandes de brevet US déposée le 19 Août 1996, la première sous le N° 08/699 295 et intitulée "System And Method For Handling Software Interrupts With Argument Passing" la seconde sous le N° 08/699 294 et intitulée "System And Method For Handling Interrupt And Exception Events In An Asymmetric Multiprocessor Architecture",toutes deux aux noms de Song et al, et dont l'ensemble est incorporé aux présentes à titre de référence.

Le registre de masque d'interruption vectoriel (VIMSK) commande le rapport des exceptions se produisant à l'intérieur du coprocesseur 204 vers le processeur 202. Les bits du VIMSK, lorsqu'ils sont réglés en même temps qu'un bit correspondant du registre de la source d'interruption vectorielle (VISRC), permet à l'exception d'interrompre le processeur 202. Le registre VISRC comprend un ensemble de bits indiquant qu'elle est la source d'un ensemble d'exceptions et d'interruptions. Les bits du registre VIMSK comprennent une autorisation d'Interruption de Point de Rupture d'Adresses de Données (DABE), une autorisation d'interruption de point de rupture d'adresses d'instructions (IABE) et une autorisation d'interruption d'un seul chemin (SSTPE). Le VIMSK commande en outre les bits d'autorisation d'interruption de dépassement de capacité (FOVE), d'opérande incorrecte (FINVE) et de division par zéro (FDIVE), à virgule flottante, ainsi que les bits d'autorisation d'interruption de dépassement de capacité (IOVE) et de division par zéro (IDIVE), entière. Le VIMSK commande également une autorisation d'interruption du

VCINT (VIE), une autorisation d'interruption du VCJOIN (VJE), une autorisation de commutation de contexte (CSE).

Le coprocesseur 204 agit en liaison avec le processeur 202 en envoyant des signaux au processeur 202. De manière spécifique, le coprocesseur 204 envoie des signaux vers le processeur 202 de manière indirecte par l'intermédiaire de registres étendus d'utilisateur indiquant que le coprocesseur 204 a exécuté une instruction de synchronisation. Le coprocesseur 203 envoie également des signaux vers le processeur 202 par l'intermédiaire d'une requête d'interruption indiquant que le coprocesseur 204 a arrêté son exécution et est entré dans l'état VP_IDLE. Le coprocesseur 204 exécute deux instructions pour envoyer des signaux au processeur 202. Une instruction VCJOIN (VCJOIN *n*) se réunit de manière conditionnelle au processeur 202 et amène le coprocesseur 204 à s'arrêter et à entrer dans l'état VP_IDLE. Un programme antagoniste (non illustré) du coprocesseur 204 adresse l'instruction qui suit l'instruction VCJOIN. L'instruction VCJOIN, exécutée par le coprocesseur 204, est classée dans une classe de déroulement de commande. Les instructions de déroulement de commande comprennent diverses instructions conditionnelles telles que des instructions de branchement, de décrément et branchement, de saut, de retour de sous-programme, de commutation de contexte et de barrière.

En se référant à nouveau à la figure 2, le sous-système d'antémémoire 208 comprend une antémémoire de données 236 (par exemple, 5 kilo-octets), une antémémoire d'instructions 238 (par exemple, 2 kilo-octets) et une ROM d'antémémoire 240 (par exemple, 16 kilo-octets) et fonctionne de manière typique à la même vitesse que le coprocesseur 204 (80 MHz). Dans un mode de réalisation, le sous-système d'antémémoire 208 comprend 1 kilo-octet de mémoire d'instructions et 1 kilo-octet de mémoire de données pour le processeur 202, 1 kilo-octet de mémoire

d'instructions et 4 kilo-octets de mémoire de données pour le coprocesseur 204, et le partage d'une ROM de 16 kilo-octets d'antémémoire d'instructions et de données intégrées pour à la fois le processeur 202 et le coprocesseur 204. Le sous-système d'antémémoire 208 fournit une interface pour le processeur 202 par l'intermédiaire de bus de données à 32 bits et une interface pour le coprocesseur 204 par l'intermédiaire de bus de données à 128 bits. La ROM d'antémémoire 240 comprend un logiciel d'initialisation de μROM, un logiciel de diagnostics d'auto-essai, divers logiciels de gestion système, des programmes de bibliothèque et une antémémoire pour des constantes d'instructions et de données choisies. De manière spécifique, la ROM d'antémémoire 240 comprend un organe de traitement d'exception d'instructions et des organes de traitement d'interruption de dispositifs entrée et sortie 0, 1, 2 et 3 pour le processeur 202. La ROM d'antémémoire 240 comprend également un organe de traitement d'interruption du processeur vectoriel et un organe de traitement d'exception de point de rupture du processeur vectoriel qui s'exécutent dans le processeur 202.

En se référant à la figure 5, un schéma synoptique illustre l'architecture de logiciel et de micro-programmation 500 du processeur de signaux multimédia 200 comprenant un logiciel d'éléments de système MSP 502 s'exécutant sur le processeur de signaux multimédia 200 et un logiciel de système d'application PC et de fonctionnement 508 s'exécutant sur le processeur central 102. Le processeur de signaux multimédia 200 est commandé par une microprogrammation comprenant une bibliothèque de microprogrammation à vecteur DSP 504 qui s'exécute sur le coprocesseur 204 et un bloc de fonctions de gestion système 506 qui s'exécute sur le processeur 202. La bibliothèque de microprogrammation à vecteur DSP 504 et le bloc de fonctions de gestion système 506 sont inclus dans le logiciel d'éléments de système MSP 502.

L'architecture 500 sépare de manière avantageuse la fonctionnalité de traitement des signaux des opérations de commande d'application centrale pour simplifier le développement du logiciel, améliore la gestion de conception du logiciel et réduit les coûts de développement et de maintenance des applications.

Le logiciel d'éléments de système MSP 502 s'exécute exclusivement sur le processeur 202 et comprend une partie résidente de système de fonctionnement en temps réel MSP 510, un module de bibliothèque multimédia 512, le bloc de fonctions de gestion système 506 et la bibliothèque de microprogrammation à vecteur DSP 504. La partie résidante en temps réel MSP 510, un sous ensemble de chez Microsoft Corp., de type MMOSA real time Kermel, est de manière typique responsable de la fourniture d'une interface pour le processeur central 102, de la gestion des ressources, du traitement du dispositif E/S et de la plupart des traitements d'interruption et d'exception. La partie résidante en temps réel MSP 510 comprend un logiciel pour fournir une interface pour des logiciels Windows™ et Windows NT™ s'exécutant dans le processeur central 102. La partie résidante en temps réel MSP 510 comprend également un logiciel pour sélectionner et décharger une microprogrammation d'application choisie à partir du processeur central 102, un logiciel pour organiser des taches pour être exécutées dans le processeur 202 et dans le processeur vectoriel 204, et un logiciel pour la gestion de ressources système du processeur de signaux multimédia 200 comprenant des dispositifs mémoire et E/S. La partie résidante en temps réel MSP 510 comprend un logiciel pour synchroniser une communication entre des taches du processeur de signaux multimédia 200 et un logiciel pour rapporter des conditions d'interruption, d'exception et d'état, reliées au MSP.

La bibliothèque de microprogrammation à vecteur DPS 504 effectue sensiblement toutes les fonctions de

traitements des signaux numériques. La bibliothèque de microprogrammation à vecteur DPS 504 commande également des interruptions spéciales spécifiques telles qu'une Interruption de Co-processeur qui est délivrée par le processeur 202 vers le processeur vectoriel 204, ou une Exception de Dépassement de Capacité d'Empilage de Matériel, qui est créée à l'intérieur du processeur vectoriel 204.

Le module de bibliothèque multimédia 512 effectue des fonctions de traitement de communications, y compris de communications de données, de communication, vidéo et audio MPEG, de communications de codage et de synthèse de parole, de communications audio compatibles SoundBlaster™ et analogues. La partie résidante en temps réel MSP 510 est un système de fonctionnement en temps réel, robuste, multitache, pré-vide, comprenant des perfectionnements qui facilitent l'exécution d'applications multimédia sur le processeur de signaux multimédia 200.

Le logiciel d'applications PC et de système de fonctionnement 508 s'exécutant dans le processeur central 102 commande le processeur de signaux multimédia 200 en lisant et en enregistrant des registres de commande et d'état MSP par l'intermédiaire du bus système 106, et en enregistrant dans des structures de données partagées celles qui sont résidantes à la mémoire système 104 et celles qui sont résidantes au processeur multimédia 200.

L'exécution du programme MSP commence avec le processeur 202 qui exécute un premier train d'exécution. Le processeur 202 peut déclencher un second train d'exécution indépendant dans le processeur vectoriel 204. Les opérations du processeur 202 et du processeur vectoriel 204 sont synchronisées par l'intermédiaire d'instructions de coprocesseur spécifiques qui fonctionnent dans le processeur 202, comprenant des instructions STARTVP, INTVP et TESTVP, et des instructions spéciales s'exécutant dans le processeur vectoriel 204, comprenant des instructions VJOIN et VINT.

Des transferts de données entre le processeur 202 et le processeur vectoriel 204 sont effectués en utilisant des instructions de déplacement de données exécutées dans le processeur 202.

Le coprocesseur 204 agit en liaison avec le processeur 202 en envoyant des signaux vers le processeur 202. De manière spécifique, le coprocesseur 204 envoie des signaux vers le processeur 202 de manière indirecte par l'intermédiaire de registres étendus d'utilisateur indiquant que le coprocesseur 204 a exécuté une instruction de synchronisation. Le coprocesseur 204 envoie également de manière directe des signaux vers le processeur 202 par l'intermédiaire d'une requête d'interruption indiquant que le coprocesseur 204 a arrêté son exécution et a entré l'état VP_IDLE. Le coprocesseur 204 exécute deux instructions pour appliquer des signaux au processeur 202, à savoir une instruction VCJOIN et une instruction VCINT, laquelle (VCINT*n*)interrompt de manière conditionnelle le processeur 202, en amenant le coprocesseur 204 à s'arrêter et à entrer dans l'état VP_IDLE. Les instructions VCINT et VCJOIN sont des instructions exécutées par le coprocesseur 204 qui sont classées dans une classe de déroulement de commande. Les instructions de déroulement de commande comprennent diverses instructions conditionnelles telles que des instructions de branchement, de décrément et branchement, de saut, de retour de sous-programme, de commutation de contexte et de barrière.

Le système à processeur multimédia 100 est décrit de manière plus détaillée dans les demandes de brevet US déposées le 19 Août 1996 sous les Nos 08/697 102 au nom de L. Nguyen, intitulée "Microprocessor Operation in a Multimedia Signal Processor", 08/699 597 au nom de L. Nguyen, intitulée "Single-Instruction-Multiple-Data Processing in a Multimedia Signal Processor", 08/697 086 au nom L. Nguyen et al., intitulée "Single-Instruction-Multiple-Data Processing Using Multiple Banks of Vector

Registers", et 08/699 585 au nom de M. Mohamed et al., intitulée "Single-Instruction-Multiple-Data Processing With Combined Scalar/Vector Operations", dont l'ensemble est incorporé aux présentes à titre de référence.

Le processeur de signaux multimédia 200 est capable de détecter des opérations multitaches qui impliquent une exécution successive de programmes. Le temps nécessaire pour commuter de contexte entre des programmes peut être réduit en diminuant la quantité des informations d'état de processeur mis en mémoire, par rapport à une mise en mémoire de type traditionnel de toutes les informations d'état de processeur, nécessaires pour restaurer avec succès le programme en cours de commutation de contexte. En réduisant la quantité des informations de processeur mises en mémoire au cours d'une commutation de contexte, les ressources peuvent être utilisées de manière plus efficace, comme indiqué ci-dessous.

La figure 6 illustre un mode de réalisation d'un déroulement 600 de processus efficace de commutation de contexte et de restauration de programme (déroulement de processus de sauvegarde de contexte/restauration 600) qui est utilisé à titre d'exemple par le système de processeur multimédia 100 et qui utilise notamment un processeur de signaux multimédia 200 choisi parmi l'un quelconque des processeurs de signaux multimédia de la famille MSP de Samsung. Le déroulement du processus de conservation de contexte/restauration 600 commence avec le bloc d'exécution de programme 602. Dans le bloc d'exécution de programme 602, le coprocesseur 204 (figure 2) exécute un programme, tel qu'un programme d'application. Le programme d'application comprend des repères intercalés par le programmeur dans tout le programme d'application de telle manière qu'un repère soit rencontré sur une base approximativement régulière. Dans ce mode de réalisation, le repère est une instruction de commutation de contexte conditionnel du coprocesseur 204 appelé VCCS. A un point quelconque au

cours de l'exécution du programme d'application, le coprocesseur 204 comprend des informations d'état de processeur associées à ce point du programme en cours d'exécution. A certains points du programme d'application, moins d'informations d'état de processeur sont nécessaires pour restaurer avec succès le programme d'application à la suite d'une commutation de contexte qu'à d'autres points du programme d'application. Le programmeur, étant au courant de ces points "d'état de processeur réduit" du programme d'application qui nécessitent de préférence la préservation d'une quantité minimale d'informations d'état de processeur au travers d'une commutation de contexte, place l'instruction VCCS à ces points d'état de processeur réduit du programme d'application. Cependant, l'intervalle entre des instructions VCCS rencontrés devrait de préférence ne pas être tellement grand pour retarder de manière notable une opération de commutation de contexte. L'intervalle dépendra visiblement du jugement du programmeur, du programme d'application, et de la vitesse d'horloge du coprocesseur 204. Un intervalle pris à titre d'exemple résulte en un retard inférieur à 2 μsec entre une commutation de contexte requise et la réponse à la requête.

La réduction de la quantité des informations d'état de processeur nécessaires pour reprendre avec succès un programme à la suite d'une commutation de contexte peut avoir pour résultat des gains appréciables de performance du processeur, notamment lorsque la dimension potentielle des informations d'état du processeur est très grand. Par exemple, l'un quelconque de la famille MSP de Samsung constitue une architecture prise à titre d'exemple présentant une quantité très grande de zones de stockage en mémoire allouée aux informations d'état de processeur. Le MSP de Samsung comprend plus de sept (7) kilo-octets d'informations d'état de processeur et comprend soixante quatre (64) registres vectoriels à deux cent quatre vingt

huit (288) bits, plus de quatre vingt (80) registres scalaires à trente deux (32) bits, et jusqu'à quatre (4) Kbytes de mémoire de travail. Les augmentations de performance associées à la réduction de la quantité du transfert des informations d'état de processeur à la fois pendant une commutation de contexte et ensuite pendant la restauration du programme dans une architecture telle que le MSP de Samsung peuvent être notables.

Un exemple d'un point d'état de processeur réduit dans un programme d'application comprend une instruction ADD qui détermine la somme de 50 arguments. Si le programme d'application est commuté en contexte avant de déterminer la somme des arguments, les informations d'état de processeur nécessaires pour restaurer ce programme d'application doivent comprendre l'ensemble des 50 arguments. Cependant, si l'instruction ADD est complète et que la somme est déterminée, seule la somme doit être incluse en tant que partie des informations d'état de processeur. Par conséquent, la quantité des informations d'état de processeur nécessaires pour la reprise du programme lors d'un retour au programme est réduite par rapport à la sauvegarde de toutes les informations d'état de processeur disponibles. L'instruction VCCS, qui constitue l'instruction de commutation de contexte conditionnelle du coprocesseur 204, est par conséquent placée dans le programme d'application à un point qui suit la détermination de la somme des 50 arguments. Ceci constitue un exemple d'un point approprié du programme d'application pour réduire les informations d'état de processeur requises afin de commuter avec succès le contexte et restaurer le programme d'application. Il apparaîtra à l'homme du métier après la lecture de cette description que de nombreux autres points des programmes d'application peuvent être choisis pour réduire la quantité de mise en mémoire d'informations d'état de processeur en prévision

de la commutation de contexte et de la restauration ultérieure d'un programme.

En se référant à la figure 7, un segment 700 pris à titre d'exemple d'un programme d'application comprend des instructions VCCS intercalées 702. Les instructions VCCS 702 sont en général intercalées de manière régulière à des emplacements prédéterminés requérant une quantité minimale de mise en mémoire d'informations d'état de processeur en prévision d'une commutation de contexte réalisée avec succès et sans séparation dans le programme d'application.

En se référant à nouveau à la figure 6, au cours de l'exécution du programme d'application dans le bloc d'exécution de programme 602, une instruction VCCS de (32) bits sera rencontrée au niveau du point approprié comme montré dans le bloc d'instructions de commutation de contexte 604. En continuant vers le bloc de requête de commutation de contexte 606, l'instruction VCCS amène le coprocesseur 204 à déterminer si le processeur 202 a ou non requis que le programme d'application en cours d'exécution soit commuté en contexte et remplacé par un autre programme. Si une commutation de contexte n'a pas été requise, le déroulement du processus de sauvegarde de contexte/restauration 600 revient au bloc d'exécution de programme 602 et continue à exécuter le programme d'application en commençant au point qui suit l'instruction VCCS rencontrée.

De manière à requérir une commutation de contexte, le processeur 202 enregistre un registre de masquage d'interruption de processeur vectoriel à trente deux (32) bits 208, appelé registre VIMSK, placé dans l'interface de coprocesseur 206. De manière spécifique, le processeur 202 envoie le bit zéro (0), qui constitue le bit CSE ou d'autorisation de commutation de contexte, du registre VIMSK 204 en enregistrant un (1) au bit CSE. Pour davantage de renseignements sur le registre VIMSK, voir les demandes de brevet US déposées le 19 Août 1997 aux

noms de Song et al respectivement sous le N° 08/699 295 et intitulée "System And Method For Handling Software Interrupts With Argument Passing" et sous le N° 08/699 294 et intitulée "System And Method For Handling Interrupt And Exception Events In An Asymmetric Multiprocessor Architecture."

Le tableau 1 énonce le format de l'instruction VCCS, utilisé dans le processeur vectoriel MSP de Samsung.

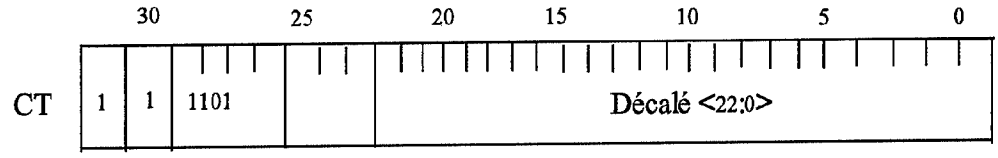


TABLEAU 1

Le format de l'instruction VCCS comprend des bits 111101 dans les positions des bits les plus significatifs et comprend une zone Décalée dans les vingt trois (23) bits les moins significatifs. La zone Décalée identifie un emplacement de sous-programme de sauvegarde de contexte auquel le coprocesseur 204 se branchera lors d'un achèvement avec succès de l'instruction DCCS si une commutation de contexte a été requise. Les bits (25:23) ne sont pas utilisés par l'instruction VCCS. La syntaxe d'assembleur de l'instruction VCCS est VCCS #Décalé.

En se référant au bloc de requête de commutation de contexte 606, afin de déterminer si le processeur 202 a ou non requis que le programme d'application en cours d'exécution dans le bloc d'exécution de programme 602 soit commuté en contexte et remplacé par un autre programme, l'instruction VCCS rencontrée amène le coprocesseur 204 à lire le registre VIMSK 214. En supposant que l'instruction VCCS n'amène pas une exception, si le bit CSE du registre VIMSK 214 est réglé à un (1), l'adresse de retour du programme d'application du bloc d'exécution de programme 602 est sauvegardée sur une pile d'adresses de retour temporaire associée de

manière unique au programme en cours de commutation de contexte. L'adresse actuelle du programme d'application est placée dans le compteur de programme vectoriel (VPC). L'adresse de retour est égale à VPC plus quatre (4).

On notera que le coprocesseur 204 ne vérifie pas une commutation de contexte requise à partir du processeur 202 jusqu'à ce qu'une instruction VCCS soit rencontrée. Il en résulte que le programmeur peut commander le point au niveau duquel un programme d'exécution peut être commuté en contexte. En outre, bien qu'il puisse exister un retard entre une commutation de contexte requise et une réponse correspondante, aucune pénalité d'exécution de programme n'est encourue du fait que le programme d'application continue à s'exécuter pendant toute requête de commutation de contexte, toute détection de commutation de contexte et tout retard de réponse de commutation de contexte.

Le pseudocode suivant énonce le fonctionnement du VCCS :

```
Si (VIMSK<CSE> == 1){
        Si (VSP<4> 15){
            VISRV<RASO>=1 ; processeur de
            signaux 202 avec exception RASO ;
            VP_STATE = VP_IDLE ;
        }autrement{
            RSTACK[VSP<3:0] = VPC + 4 ;
            VSP<4:0> = VSP<4:0> + 1 ;
            VPC = VPC + sex(Décalé<22:0>*4) ;
        }
}autrement VPC = VPC + 4 ;
```

En se référant au bloc de requête de commutation de contexte 606, l'instruction VCCS amène ainsi le coprocesseur 204 à déterminer si le bit CSE du registre VIMSK 214 est ou non réglé. Si le bit n'est pas réglé, le processeur 202 n'a pas requis une commutation de contexte, et le compteur de programme (VPC) du coprocesseur 204 est incrémenté à l'instruction suivante

et un retour est effectué au bloc d'exécution de programme 602. Si le processeur 202 a requis une commutation de contexte, le coprocesseur 204 examine le bit quatre (4) du repère (VSP) de la pile vectorielle de cinq (5) bits associé au programme d'application et détermine si l'exécution de l'instruction VCCS aura pour résultat un débordement de capacité de la pile. Si un débordement de capacité de pile se produit, le bit d'exception de Débordement de Capacité de Pile d'Adresses de Retour du Registre de Source d'Interruption Vectorielle (VISRC) est réglé à un (1), et l'état (VP-STATE) du coprocesseur (204) est placé dans un état inactif (VP_IDLE). Si un débordement de capacité de pile ne se produit pas, le déroulement du processus de sauvegarde de contexte/restauration 600 avance pour sauvegarder le bloc d'adresse de retour 608, et le coprocesseur 204 sauvegarde l'adresse de retour du programme d'application qui s'exécute dans le bloc d'exécution de programme 602 et incrémente le repère de pile temporaire. VPC est alors chargé avec l'adresse étendue du signe indiquée par la zone de Décalée VCCS. L'adresse maintenant stockée dans VPC est l'adresse du sous-programme de sauvegarde de contexte montrée dans le bloc de sous-programme de sauvegarde de contexte 612.

Le coprocesseur 204 se branche ensuite vers le sous-programme de sauvegarde de contexte en utilisant l'adresse de sous-programme de sauvegarde de contexte du VPC comme montrée dans le branchement d'exécution vers le bloc de sous-programme de sauvegarde de contexte 610. Le sous-programme de sauvegarde de contexte 612 est enregistré par le programmeur du programme d'application exécuté dans le bloc d'exécution de programme 602. Il en résulte que le programmeur est au courant de la quantité minimale d'informations d'état de processeur qui doit nécessairement être mise en mémoire afin de reprendre avec succès le programme d'application à commutation de contexte. Comme montré dans le sous-bloc d'informations

d'état de processeur minimal de mise en mémoire 614, le sous-programme de sauvegarde de contexte 612 ne sauvegarde que les informations requises présentes, par exemple, dans les registres et la mémoire de travail, qui seront nécessaire pour restaurer le programme à commutation de contexte.

En avançant vers l'emplacement de sauvegarde du sous-bloc 616 du sous-programme de restauration de contexte, le coprocesseur 204 termine le sous-programme de sauvegarde de contexte 612 en exécutant une instruction de tache de Liaison Conditionnelle de trente deux (32) avec le processeur 202, instruction appelée VCJOIN. Le Tableau 2 énonce le format de l'instruction VCJOIN utilisé dans le processeur vectoriel MSP de Samsung.

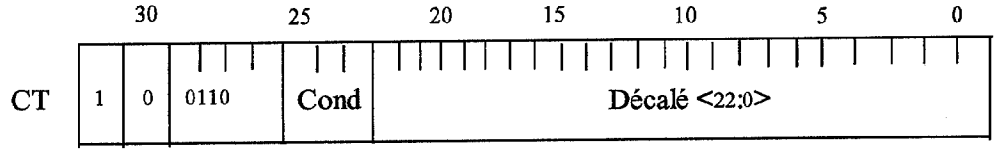


TABLEAU 2

Le format de l'instruction VCJOIN inclut des bits 100110 dans les positions des bits les plus significatifs et comprend une zone de condition dans les bits (25:23) et une zone Décalée dans les vingt trois (23) bits les moins significatifs. La zone de condition est réglée pour interrompre de manière inconditionnelle le processeur 202. La zone Décalée identifie une adresse d'un emplacement du sous-programme de restauration de contexte que le coprocesseur 204 exécutera lors de la reprise du programme à commutation de contexte. La syntaxe d'assembleur de l'instruction VCJOIN est VCJOIN.cond#Décalé.

Le pseudocode suivant énonce le fonctionnement de VCJOIN :

```
Si (Cond = in){
        VISRC<VJP>==1 ;
```

```
VIINS = [Instruction CJOIN.cond#Décalé];
VEPC = VPC ;
     Si (VIMSK<VJE>=1;interruption du
     processeur de signaux 202 ;
VP_STATE=VP_IDLE ;
}
```

autrement VPC = VPC + 4 ;

En se référant à l'emplacement de sauvegarde du sous-bloc 616 du sous-programme de restauration de contexte, l'exécution de l'instruction VCJOIN.en #Décalé par le coprocesseur 204 amène l'emplacement du sous-programme de restauration de contexte 628 à être conservé et interrompt le processeur 202 de sorte que le processeur 202 peut régler le coprocesseur 204 afin d'exécuter un programme ultérieur. Dans l'emplacement de sauvegarde du sous-bloc 616 du sous-programme de restauration de contexte, la condition VCJOIN est réglée à inconditionnel, et un bit (VJE) d'Autorisation d'Interruption de VCJOIN du registre VIMSK 214 est également réglé à (1). Le bit VJE est le bit cinq (5) du registre VIMSK 214. En se référant à l'emplacement de sauvegarde du sous-bloc 616 du sous-programme de restauration de contexte, la zone Décalée de vingt trois (23) bits de l'instruction VCJOIN est réglée pour indiquer un emplacement du sous-programme de restauration de contexte 628.

Lorsque l'instruction CJOIN est exécutée par le coprocesseur 204 dans le sous-bloc 618 du processeur d'interruption, le coprocesseur 204 contrôle la zone de condition de l'instruction VCJOIN pour vérifier l'état inconditionnel de la zone de condition. Ensuite, le coprocesseur 204 règle le bit d'exception en cours (VJP) du Registre de Source d'Interruption Vectorielle à trente deux (32) bits (VISRC) à un (1). Le bit VJP est le bit cinq (5) du registre VISRC. Un Registre d'Instructions d'Interruption Vectoriel 1 (VIINS) est mis à jour avec l'instruction VCJOIN car l'instruction VCJOIN est en

cours d'exécution afin d'interrompre le processeur 202. Le VPC en cours est mis en mémoire dans un Compteur de Programme d'Exception Vectoriel (VEPC). Le VEPC spécifie l'adresse de l'instruction qui a le plus probablement causé l'exception la plus récente. Comme l'exécution de l'instruction VCJOIN continue, le bit VJE du registre VIMSK 214 est évalué et déterminé pour être réglé à un (1). Le réglage du bit VJE amène une interruption IRQ du processeur 202, et le réglage du bit VJP amène le coprocesseur 204 à entrer dans l'état IDLE.

Le déroulement du processus de sauvegarde de contexte/restauration 600 désélectionne le bloc 612 du sous-programme de sauvegarde de contexte et avance vers le bloc 620 de l'organe de traitement d'interruption de processeur. Le processeur 202 traite l'interruption comme décrit dans les demandes de brevet US déposées le 19 Août 1996 aux noms de Song et al, d'une part sous le N° 08/699 295 et intitulé "System And Method For Handling Software Interrupts With Argument Passing" et, d'autre part, sous le N° 08/699 294 et intitulé "System And Method For Handling Interrupt And Exception Events In An Asymmetric Multiprocessor Architecture."

Après traitement de l'interruption déclenchée par VCJOIN, le processeur 202, comme montré dans le bloc 622 de décision de commutation de contexte du programme suivant, détermine si le programme suivant à exécuter est ou non un programme "nouveau", c'est-à-dire un programme dont le contexte n'a pas été précédemment commuté ou un programme dont le contexte a été précédemment commuté. Le système de fonctionnement (figure 5) du processeur de signaux multimédia 200 garde trace des programmes et du fait que leur contexte a été ou non précédemment commuté. Si le programme suivant à exécuter par le processeur 204 est "nouveau", le déroulement du processus de sauvegarde de contexte/restauration 600 avance vers le coprocesseur configuré de manière à exécuter le bloc 624 du "nouveau" programme. Le processeur 202 configure le coprocesseur

204 pour exécuter le "nouveau" programme, et le déroulement du processus de sauvegarde de contexte/restauration 600 retourne vers le bloc d'exécution de programme 602 où est exécuté le "nouveau" programme.

En revenant vers le bloc 622 de décision de commutation de contexte de programme suivant, le coprocesseur 204 entre le branchement vers le bloc de sous-programme de restauration de contexte 626 si le programme suivant a déjà eu son contexte commuté. Le processeur 202 charge alors VPC du coprocesseur 204 avec l'adresse du sous-programme de restauration de contexte 628, lequel a été sauvegardé pendant l'exécution de l'instruction VCJOIN dans l'emplacement de sauvegarde du sous-bloc de sous-programme de restauration de contexte 616 comme décrit ci-dessus, et le déroulement du processus de sauvegarde de contexte/restauration 600 se branche vers le sous-programme de restauration de contexte 628.

Le sous-programme de restauration de contexte 628 est enregistré par le programmeur du programme d'application dont le contexte est en train maintenant d'être commuté. Il en résulte que le programmeur est au courant de l'emplacement de la quantité minimale d'informations d'état de processeur qui a été précédemment mise en mémoire par le sous-programme de sauvegarde de contexte 612 en prévision de la reprise avec succès de la commutation de contexte et du contexte maintenant commuté dans le programme d'application. Le déroulement du processus de sauvegarde de contexte/restauration 600 avance pour charger le sous-bloc d'informations d'état de processeur minimal 630 où le coprocesseur 204 avance pour conduire la quantité minimale précédemment stockée des informations d'état de processeur dans les emplacements mémoire appropriés du coprocesseur 204, par exemple dans les registres et la mémoire de travail précédemment décrits.

Ayant chargé les informations d'état de processeur nécessaires pour la reprise avec succès de la commutation de contexte dans le programme d'application, le sous-programme de restauration de contexte 628 prépare le coprocesseur 204 à exécuter le programme d'application à commutation de contexte au niveau de l'emplacement de programme précis qui a été précédemment sauvegardé dans l'emplacement de sauvegarde du sous-bloc 616 du sous-programme de restauration de contexte. Afin de restaurer cet emplacement de programme d'application, le coprocesseur 204 exécute une instruction de Retour Conditionnel De Sous-programme (VCRSR) qui termine le sous-programme de restauration de contexte 628. Le Tableau 3 énonce le format de l'instruction VCJOIN qui est utilisée dans le processeur vectoriel MSP de Samsung.

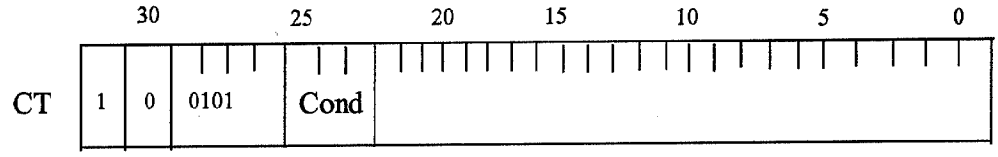


TABLEAU 3

Le format de l'instruction VCRSR comprend les bits 100101 dans les positions de bits les plus significatifs et comprend une zone de condition dans les bits (25:23). Les vingt trois (23) bits les moins significatifs ne sont pas utilisés et peuvent être réglés à une valeur quelconque. La zone de condition est réglée pour interrompre de manière inconditionnelle le processeur 202. La synthaxe d'assembleur de l'instruction VCRSR est VCRSR.cond.

Le pseudocode suivant énonce le fonctionnement de l'instruction VCRSR :

```
Si (Cond = in){
        Si (VSP<4:0>==0){
            VISRV<RASU>=1 ;
            processeur de signaux 202 avec
            exception RASU ;
```

```
                    VP_STATE = VP_IDLE ;
               }autrement{
                    VSP<4:0> = VSP<4:0> - 1 ;
                    VPC = RSTACK[VSP<3:0>] ;
                    VPC<1:0>=b'00 ;
               }
          }autrement VPC = VPC + 4 ;
```

En se référant à l'adresse de retour de charge du bloc de programme à commutation de contexte 632, l'exécution de l'instruction VCRSR.in amène le coprocesseur 204 à reprendre l'exécution de ce programme à commutation de contexte depuis l'emplacement précédemment mis en mémoire dans le bloc d'adresses de retour de sauvegarde 608. Pendant l'exécution de l'instruction VCRSR.in, le coprocesseur 204 détermine qu'un branchement inconditionnel est requis en examinant la zone de condition de l'instruction VCRSR qui contient un code de branchement inconditionnel. Le VSP examiné du fait de l'exécution de l'instruction VCRSR est un indicateur d'empilage temporaire qui est associé de manière unique au programme en cours de commutation de contexte. Le coprocesseur 204 examine ensuite le VSP pour déterminer si VSP est ou non en train de pointer vers son emplacement le moins significatif. Si le VSP est en train de pointer vers son emplacement le moins significatif, le bit RASU du registre VISRC est réglé à un (1). Le bit RASU est le bit d'exception de Dépassement de Capacité d'Empilage d'Adresse de Retour et, lorsqu'il est réglé, il signale au processeur 202 qu'une exception existe, et le coprocesseur 204 entre dans l'état inactif. Si le VSP n'est pas en train de pointer vers son emplacement le moins significatif, le coprocesseur 204 décrémente le VSP de un (1) emplacement et charge le VPC avec l'adresse mise en mémoire à cet emplacement VSP choisi. Cet emplacement du VSP contient l'adresse de retour prélablement mise en mémoire dans le bloc d'adresses de retour de sauvegarde 608. Le pas VPC<1:0> = b'00 assure

que les deux (2) bits les moins significatifs du VPC sont chargés avec des zéros.

Après que l'adresse de retour soit chargée dans le VPC, le déroulement du processus de commutation de contexte 600 retourne vers le bloc d'exécution de programme 602 où le programme maintenant à commutation de contexte commence son exécution. Le déroulement du processus de commutation de contexte 600 se répète jusqu'à ce que l'exécution du programme soit terminée.

Comme cela est évident d'après ce qui précède, l'efficacité de la commutation de contexte dans un environnement à multiprocesseurs multitache est amélioré par le déroulement de processus de commutation de contexte 600. En sauvegardant seulement la quantité minimale d'informations d'état de processeur pendant la commutation de contexte d'un programme, un temps précieux du coprocesseur 204 peut être utilisé pour d'autres opérations. En outre, le fait de restaurer seulement une quantité minimale d'informations d'état de processeur permet à un programme d'être efficacement à commutation de contexte. Ces économies de temps s'accumulent pendant que sont effectuées les opérations de commutation de contexte.

L'invention n'est pas limitée aux exemples de réalisation représentés et décrits en détail car diverses modifications peuvent y être apportées sans sortir de son cadre. Par exemple, les réalisations spécifiques de matériel et de logiciel mentionnées constituent de simples exemples et de nombreuses autres architectures de système et/ou modes de réalisation de logiciel peuvent mettre en oeuvre les caractéristiques de commutation de contexte et de restauration décrites ici. En outre, l'homme du métier comprendra, après lecture de la description ci-dessus, que les caractéristiques de commutation de contexte et de restauration peuvent être utilisées dans des environnements multitache impliquant plus de deux taches.

REVENDICATIONS

1. Procédé caractérisé en ce qu'il comprend les opérations consistant à :

exécuter un premier programme sur un premier processeur ;

recevoir une requête de commutation de contexte en provenance d'un second processeur ; et

répondre à la requête de commutation de contexte au niveau d'un point approprié du premier programme, où le point approprié est associé à un repère du programme qui indique un point approché du premier programme requérant une quantité minimale de mise en mémoire d'informations d'état de processeur en vue d'une restauration avec succès du programme.

2. Procédé selon la revendication 1 caractérisé en ce qu'il comprend en outre les opérations consistant à :

intercaler un ensemble de repères dans le premier programme au niveau d'emplacements requérant une quantité minimale de mise en mémoire d'informations d'état de processeur pour une restauration avec succès du premier programme.

3. Procédé selon la revendication 1 caractérisé en ce qu'il comprend en outre les opérations consistant à :

sauvegarder les informations d'état de processeur correspondant à un état du premier programme au niveau du point approprié ;

commuter le premier programme ;

exécuter un second programme sur le premier processeur ;

commuter le second programme ; et

restaurer le premier programme en utilisant la quantité minimale d'informations d'état de processeur mises en mémoire.

4. Procédé selon la revendication 1 caractérisé en ce que l'opération de réponse comprend les opérations consistant à :

sauvegarder les informations d'état de processeur correspondant à un état du premier programme au niveau d'un point approprié ; et

sauvegarder l'emplacement d'un programme de restauration de contexte pour restaurer sans séparation le premier programme.

5. Procédé selon la revendication 1 caractérisé en ce que l'opération d'exécution comprend une opération consistant à :

exécuter le premier programme sur un processeur vectoriel d'un processeur de signaux multimédia dans un système à multiprocesseurs multimédia.

6. Procédé selon la revendication 1 caractérisé en ce qu'il comprend en outre les opérations consistant à :

recevoir des données d'un ensemble de dispositifs multimédia ; et

en ce que l'opération d'exécution comprend l'opération consistant à traiter les données reçues.

7. Procédé selon la revendication 1 caractérisé en ce que l'opération de réception comprend l'opération consistant à :

lire un registre présentant une zone d'autorisation de commutation de contexte réglée par le second processeur.

8. Procédé selon la revendication 1 caractérisé en ce qu'il comprend en outre l'opération consistant à :

continuer d'exécuter le premier programme après avoir reçu la requête de commutation de contexte ;

et en ce que l'opération de réponse comprend les sous-opérations consistant à :

rencontrer le repère du premier programme au niveau du point approprié ;

interrompre le premier processeur ;

lire la requête de commutation de contexte avec le premier processeur en réponse à la rencontre du repère du premier programme ;

mettre en mémoire une adresse de retour du premier programme ;

mettre en mémoire la quantité minimale d'informations d'état de processeur requises pour restaurer avec succès le premier programme ; et

notifier le second processeur de la capacité du premier processeur à exécuter le second programme.

9. Procédé selon la revendication 1 caractérisé en ce que le repère est une instruction de commutation de contexte conditionnelle.

10. Système de calcul caractérisé en ce qu'il comprend :

un premier processeur dans un environnement multitache pour exécuter des programmes présentant des ensembles respectifs de repères de commutation de contexte intercalés ;

une première mémoire, couplée au premier processeur, affectée à la mise en mémoire des informations d'état de processeur ;

une seconde mémoire couplée au premier processeur ;

un détecteur de requête de commutation de contexte fonctionnant sur le premier processeur pour détecter, après que le processeur rencontre un des repères dans un programme d'exécution, une requête de commutation de contexte du programme ; et

un module de sauvegarde de contexte fonctionnant sur le premier processeur pour répondre à une requête de commutation de contexte détectée en sauvegardant, dans la seconde mémoire, les informations d'état de processeur placées dans la première mémoire.

11. Système de calcul selon la revendication 10 caractérisé en ce que le module de sauvegarde de contexte répond en outre à une commutation de contexte détectée en

sauvegardant un emplacement d'un module de restauration de contexte.

12. Système de calcul selon la revendication 10 caractérisé en ce qu'il comprend en outre :

un second processeur relié de manière mutuelle au premier processeur, les premier et second processeurs présentant des caractéristiques asymétriques, le second processeur comprenant un mécanisme pour requérir que le premier processeur effectue une commutation de contexte d'un programme d'exécution.

13. Dispositif de calcul selon la revendication 12 caractérisé en ce qu'il comprend en outre :

une unité d'interface couplée entre le premier et le second processeur, l'unité d'interface comprenant un registre accessible de manière mutuelle par les premier et second processeurs ;

le second processeur étant capable d'enregistrer le registre afin d'indiquer une requête de commutation de contexte ; et

le premier processeur étant capable de lire le registre pour détecter une requête de commutation de contexte provenant du second processeur.

14. Système de calcul selon la revendication 12 caractérisé en ce que :

le second processeur est un processeur de commande ; et

le premier processeur est un processeur vectoriel.

15. Dispositif de calcul selon la revendicatino 10 caractérisé en ce qu'il comprend en outre :

un module de restauration de contexte fonctionnant sur le premier processeur pour restaurer un programme de commutation de contexte en ramenant vers la première mémoire les informations d'état de processeur sauvegardées placées dans la seconde mémoire associée au programme à commutation de contexte.

16. Système de calcul selon la revendication 1 caractérisé en ce que les repères sont généralement

intercalés de manière régulière dans tout le programme au niveau d'emplacements qui nécessitent un minimum d'informations d'état de processeur pour restaurer avec succès le programme.

17. Système de calcul selon la revendication 10 caractérisé en ce que les repères sont des instructions de commutation de contexte conditionnel.

18. Procédé de sauvegarde de contexte efficace dans un environnement de système de calcul à multiprocesseurs multitache, comprenant les opérations consistant à :

insérer des instructions de commutation de contexte conditionnel dans un programme ;

exécuter le programme ;

recevoir une requête de commutation de contexte ;

détecter une des instructions de commutation de contexte conditionnel ;

déterminer si la requête de commutation de contexte existe à partir d'un processeur à la suite de l'opération de détection d'une des instructions de commutation de contexte conditionnel ; et

commuter le programme comprenant les sous-opérations consistant à :

sauvegarder une adresse de retour du programme lors de la détermination de l'existence d'une requête de commutation de contexte, et autrement continuer d'exécuter le programme ; et

exécuter un module de sauvegarde de contexte, comprenant les sous-opérations consistant :

mettre en mémoire des informations d'état de processeur correspondant à l'état du programme avant la détection de l'instruction de commutation de contexte conditionnel ;

sauvegarder un emplacement du module de restauration de contexte ; et

interrompre le processeur.

19. Procédé selon la revendication 18 caractérisé en ce qu'il comprend en outre les opérations consistant à :

commuter le programme comprenant les sous-opérations consistant à :

exécuter un module de restauration de contexte, comprenant les sous-opérations consistant à :

a) charger l'information d'état de processeur préalablement mise en mémoire dans l'opération d'exécution de module de sauvegarde de contexte ; et

b) charger le retour sauvegardé du programme préalablement sauvegardé dans l'opération de commutation de programme ; et

exécuter le programme.

20. Procédé selon la revendication 10 caractérisé en ce qu'il comprend en outre les opérations consistant à :

exécuter le programme sur un coprocesseur ; et

soumettre la requête de commutation de contexte provenant d'un processeur de commande à une unité d'interface.

21. Procédé selon la revendication 18 caractérisé en ce que les instructions de commutation de contexte conditionnel dans un programme comprennent les opérations consistant à :

insérer les instructions de commutation de contexte conditionnel de manière approximativement régulière dans tout le programme de sorte que les instructions de commutation de contexte conditionnel sont insérées au niveau d'emplacements du programme qui requièrent une quantité minimale de préservation des informations d'état de processeur en prévision de la commutation de contexte du programme.

22. Procédé selon la revendication 21 caractérisé en ce que l'opération d'insertion comprend en outre l'opération consistant à :

insérer les instructions de commutation de contexte conditionnel au niveau d'intervalles approximativement réguliers de manière que des retards notables soient évités entre les opérations de réception de la requête de commutation de contexte et la détection d'une des instructions de commutation de contexte conditionnel.

23. Procédé selon la revendication 18 caractérisé en ce que l'exécution d'une opération de programme comprend l'exécution du programme sur un premier processeur, le procédé comprenant en outre l'opération consistant :

interrompre de manière approximativement régulière le premier processeur lors de la détection d'une des instructions de commutation de contexte conditionnel.

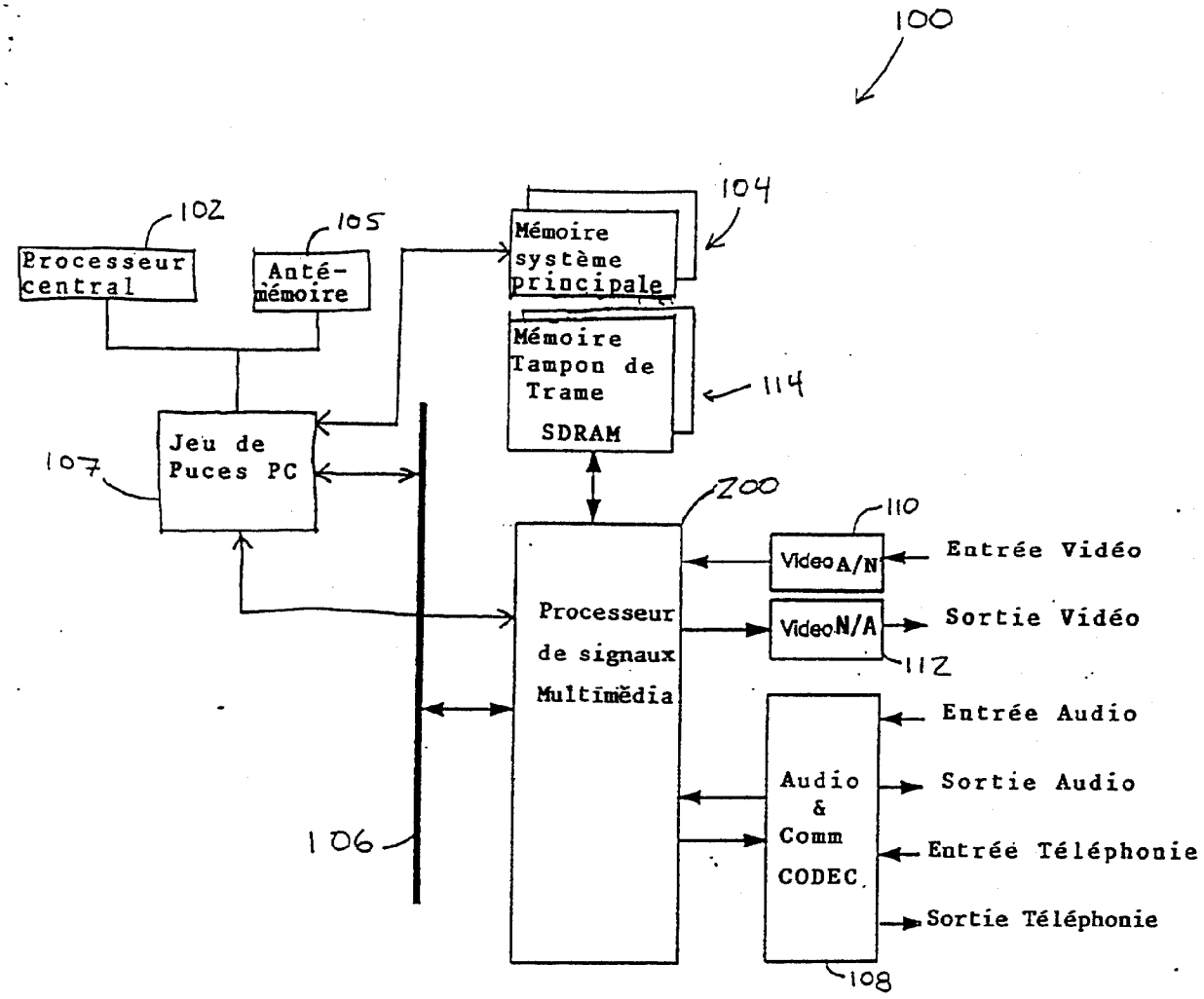

100
102
Processeur central
105
Anté-mémoire
104
Mémoire système principale
Mémoire Tampon de Trame SDRAM
114
Jeu de Puces PC
107
200
110
Video A/N
Entrée Vidéo
Processeur de signaux Multimédia
Video N/A
Sortie Vidéo
112
Entrée Audio
Audio & Comm CODEC
Sortie Audio
Entrée Téléphonie
Sortie Téléphonie
106
108

FIG. 1

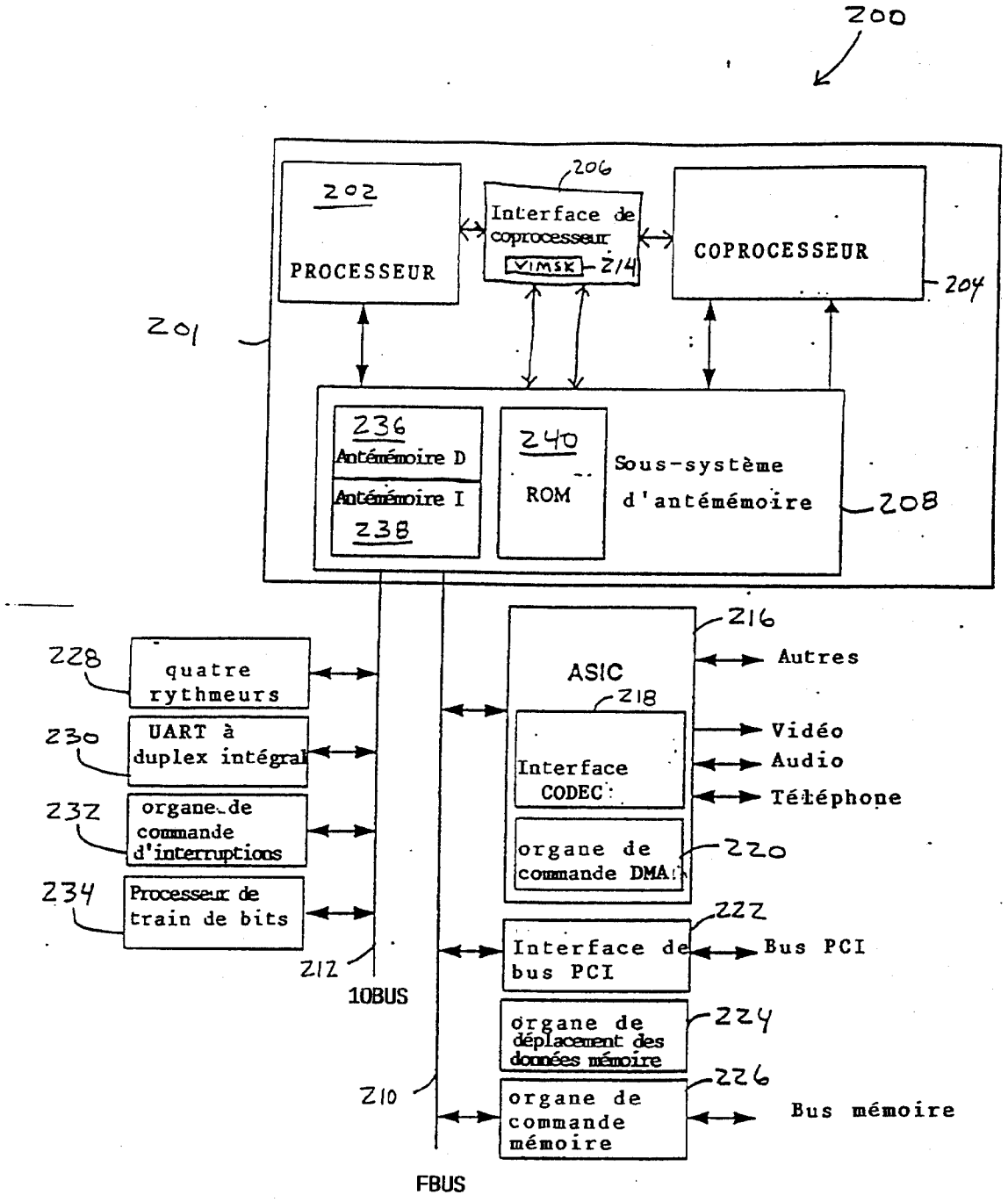

200
202
PROCESSEUR
206
Interface de coprocesseur
VIMSK
214
COPROCESSEUR
204
201
236
Antémémoire D
Antémémoire I
238
240
ROM
Sous-système d'antémémoire
208
216
ASIC
218
Interface CODEC
Autres
Vidéo
Audio
Téléphone
organe de commande DMA
220
228
quatre rythmeurs
230
UART à duplex intégral
232
organe de commande d'interruptions
234
Processeur de train de bits
212
10BUS
222
Interface de bus PCI
Bus PCI
224
organe de déplacement des données mémoire
226
organe de commande mémoire
Bus mémoire
210
FBUS

FIG. 2

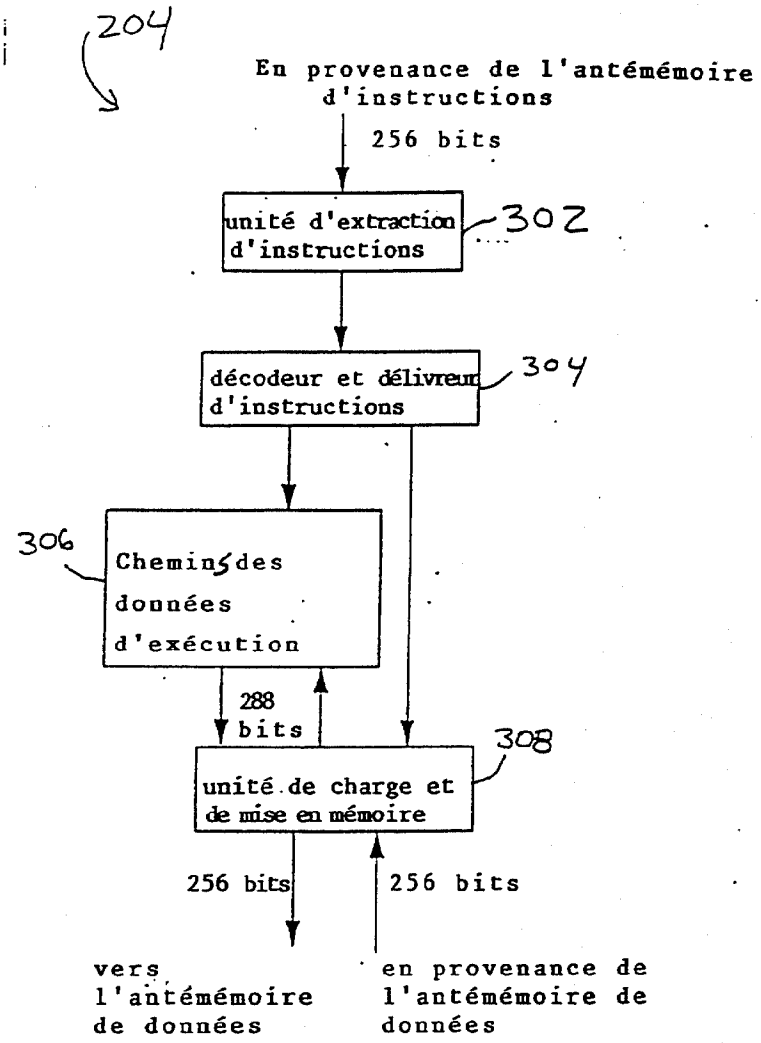

204
En provenance de l'antémémoire d'instructions
256 bits
unité d'extraction d'instructions
302
décodeur et délivreur d'instructions
304
306
Chemins des données d'exécution
288 bits
308
unité de charge et de mise en mémoire
256 bits
256 bits
vers l'antémémoire de données
en provenance de l'antémémoire de données

FIG. 3

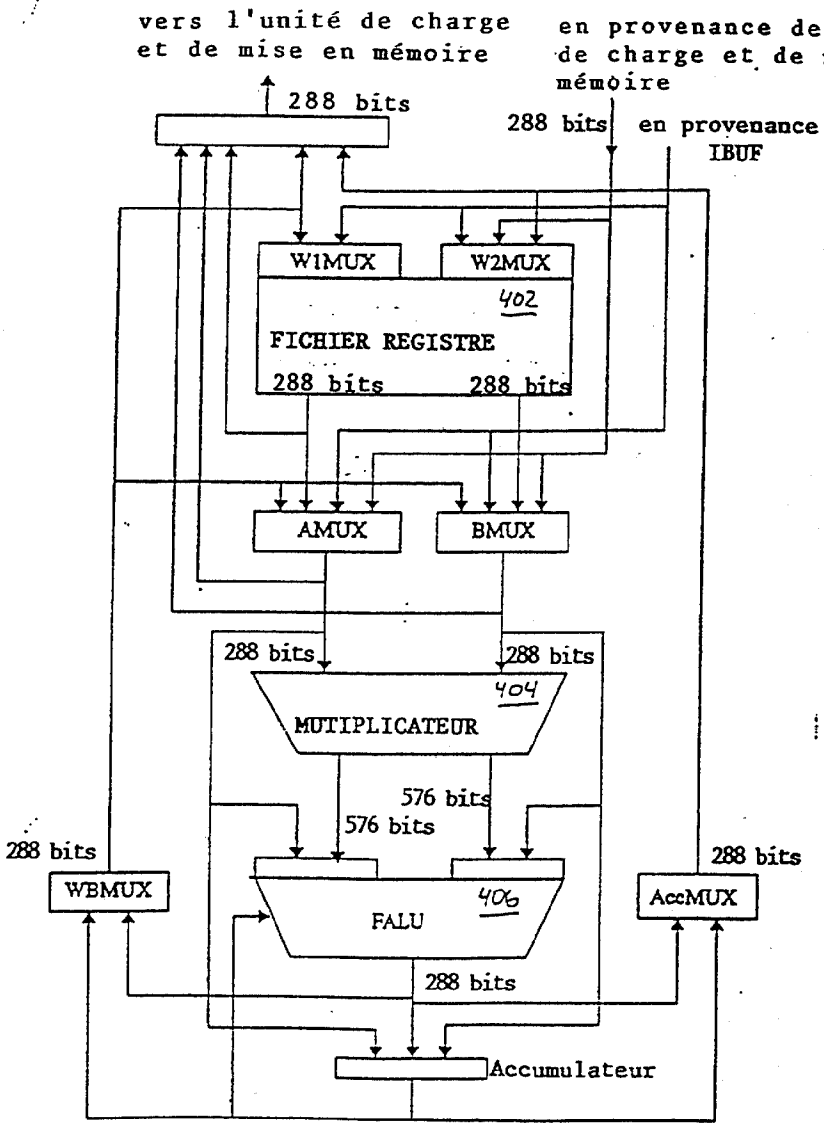

vers l'unité de charge et de mise en mémoire
en provenance de l'unité de charge et de mise en mémoire
288 bits
288 bits
en provenance de IBUF
W1MUX
W2MUX
402
FICHIER REGISTRE
288 bits
288 bits
AMUX
BMUX
288 bits
288 bits
404
MUTIPLICATEUR
576 bits
576 bits
288 bits
288 bits
WBMUX
406
FALU
AccMUX
288 bits
Accumulateur

FIG. 4

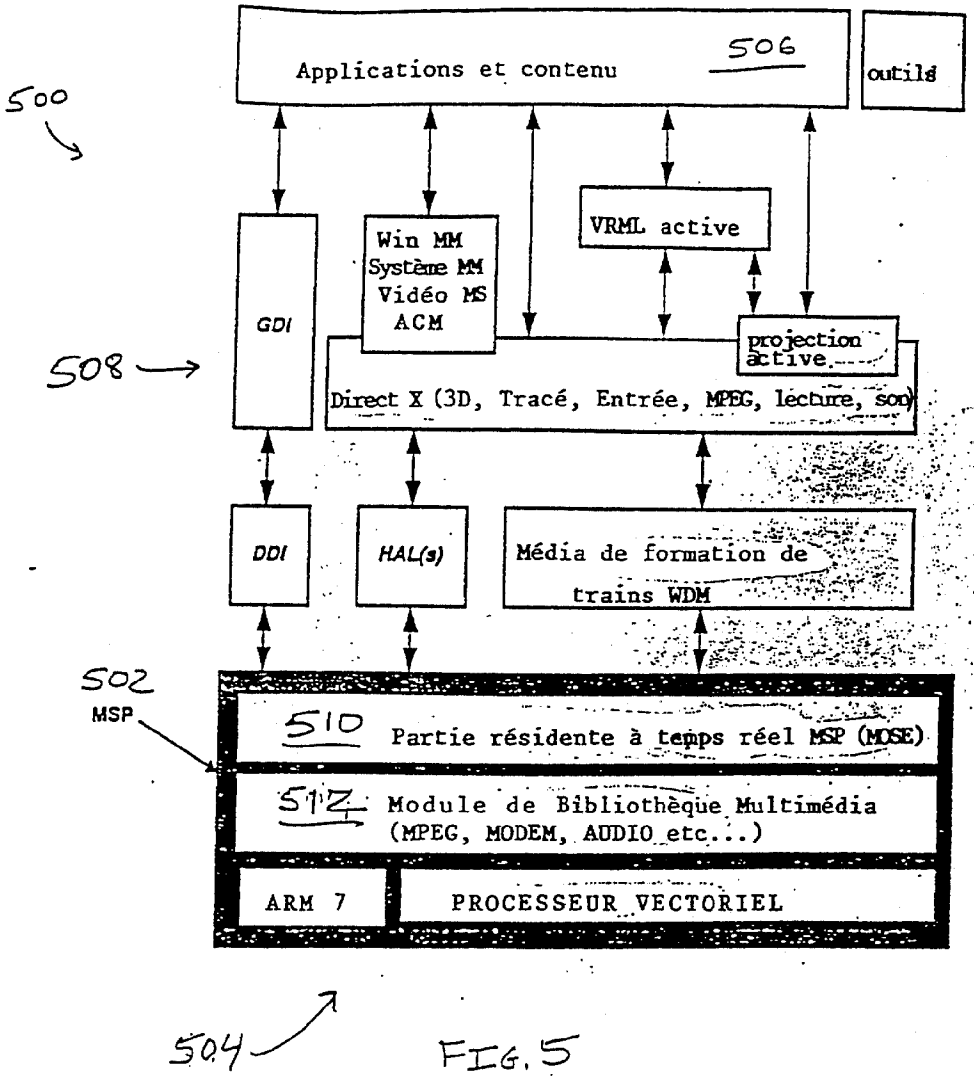

500
Applications et contenu
506
outils
GDI
Win MM
Système MM
Vidéo MS
ACM
VRML active
projection active
508
Direct X (3D, Tracé, Entrée, MPEG, lecture, son)
DDI
HAL(s)
Média de formation de trains WDM
502
MSP
510
Partie résidente à temps réel MSP (MOSE)
512
Module de Bibliothèque Multimédia (MPEG, MODEM, AUDIO etc...)
ARM 7
PROCESSEUR VECTORIEL
504

FIG. 5

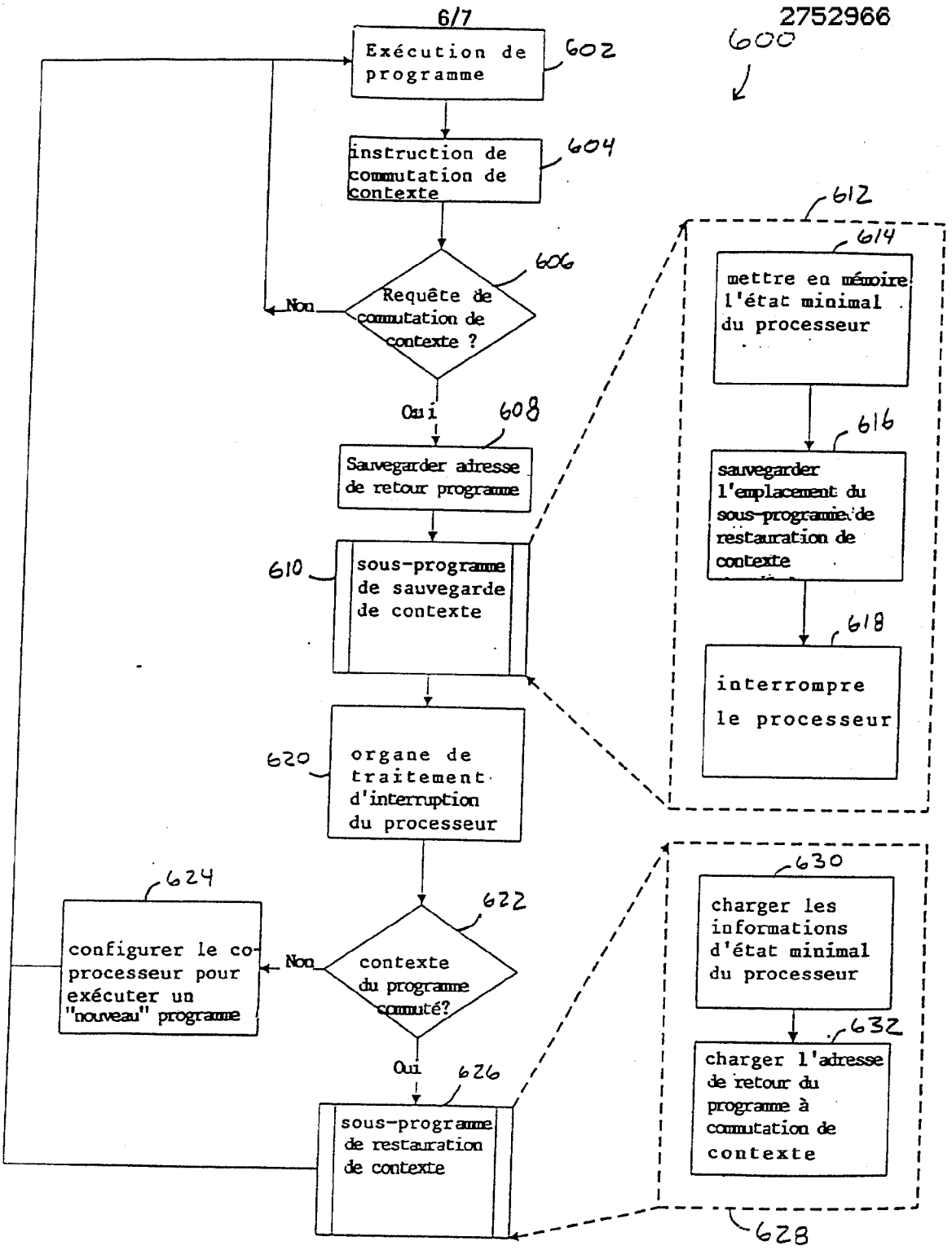

600
Exécution de programme
602
instruction de commutation de contexte
604
Requête de commutation de contexte ?
606
Non
Oui
Sauvegarder adresse de retour programme
608
sous-programme de sauvegarde de contexte
610
organe de traitement d'interruption du processeur
620
contexte du programme commuté?
622
Non
configurer le co-processeur pour exécuter un "nouveau" programme
624
Oui
sous-programme de restauration de contexte
626
612
mettre en mémoire l'état minimal du processeur
614
sauvegarder l'emplacement du sous-programme de restauration de contexte
616
interrompre le processeur
618
charger les informations d'état minimal du processeur
630
charger l'adresse de retour du programme à commutation de contexte
632
628

FIG. 6

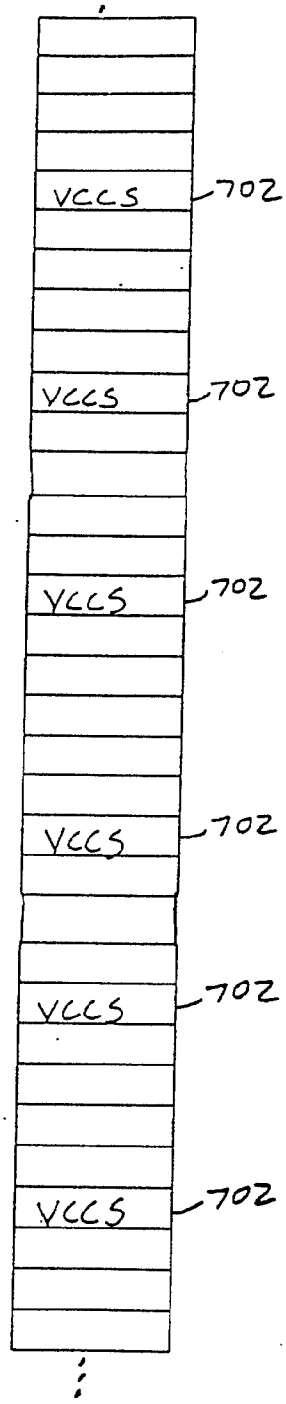


FIG. 7